Microtec
Research Inc.

Release Notes
100672-015

Microtec Research MCC68K

ANSI C Compiler

Release 4.2D

March 13, 1992

TRADEMARKS

Apollo®, AEGIS®, and DOMAIN® are registered trademarks of Apollo Computer Inc., a subsidiary of Hewlett-Packard.

Flexible License Manager™ is a trademark of Highland Software, Inc.

High C® is a registered trademark of Metaware, Inc.

HP™ and HP-UX™ are trademarks of Hewlett-Packard Company.

IBM® is a registered trademark of International Business Machines, Inc.

ICE® is a registered trademark of Intel Corporation.

Microtec® and Paragon® are registered trademarks of Microtec Research, Inc.

MRI ASM™, MRI C™, MRI FORTRAN™, MRI Pascal™, MRI XRAY™, Source Explorer™, XHM302™, XRAY™, XRAY Debugger™, XRAY In-Circuit Debugger™, and XRAY In-Circuit Debugger Monitor™, XRAY29K™, XRAY51™, XRAY68K™, XRAY86™, XRAY80™, XRAY180™, XRAYZ80™, XRAY88K™, XRAY960™, XRAYG32™, XRAYH83™, XRAYH85™, XRAYM77™, XRAYSP™, and XRAYTX™ are trademarks of Microtec Research, Inc.

MS-DOS™ is a trademark of Microsoft Corporation.

pSOS™ is a trademark of Software Components Group.

Sun™ is a trademark of Sun Microsystems, Inc.

Tektronix® is a registered trademark of Tektronix, Inc.

UNIX® is a registered trademark of UNIX System Laboratories, Inc.

VAX™, VMS™, and ULTRIX™ are trademarks of Digital Equipment Corporation.

VRTX™ is a trademark of Ready Systems, Inc.

Zilog® is a registered trademark of Zilog Corporation.

302bug™ is a trademark of Motorola, Inc.

RESTRICTED RIGHTS LEGEND

Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subdivision (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARS 52.227-7013 of DOD Regulation 27-401, or comparable clause, as such may be amended or renumbered from time to time.

MICROTEC RESEARCH, INCORPORATED
2350 MISSION COLLEGE BLVD.
SANTA CLARA, CA 95054

© Copyright 1987, 1988, 1989, 1990, 1991, 1992 Microtec Research, Inc. All rights reserved. No part of this publication may be reproduced, transmitted, or translated, in any form or by any means, electronic, mechanical, manual, optical or otherwise, without prior written permission of Microtec Research, Inc.

Table of Contents

Documentation Errata 1

Installation 3

Flexible Licensing, Version 2.0 3

Name Expansion is Not Implemented 3

Undocumented Features in Flexible Licensing 4

New Environment Variable: LICENSE_LOG_FILE 4

Restart Script Can Send Electronic Mail 4

Known Problems in Flexible Licensing 4

Temporary Loss of Licenses When Daemons Restart 4

Avoid Relative Paths 4

MS-DOS Host 4

DOS/16M Memory Management System 4

File Placement 5

UNIX 5

MS-DOS 5

VMS 5

Search Paths 6

Release 4.2D 7

Enhancements in 4.2D 7

Corrections in 4.2D 7

Release 4.2C 8

Corrections in 4.2C 8

Release 4.2B 9

Corrections in 4.2B 9

Release 4.2A 10

Operational Changes in 4.2A 10

Enhancements in 4.2A 11

Corrections in 4.2A 13

Using MCC68K 4.2X with XRAY68K prior to 2.2 16

Known Problems 17

Microtec Research MCC68K ANSI C Compiler

Release 4.2D

Introduction

This document describes the operational changes, enhancements, problem corrections, and known problems for the MCC68K C Compiler and documentation.This information covers changes from release 4.1E through release 4.2D.

Please fill out your warranty registration card and return it to Microtec Research. Thank you.

Documentation Errata

1. The '_WARNING_xxx_stub_used' feature is discussed in the *Library Functions* chapter of the *Reference Manual*, but it is not yet implemented in the run-time libraries.

2. The following options have been added to the compiler, but are not discussed in the *User's Guide*:

VMS	UNIX/MS-DOS	
Diagnostics_to=stderr	-Fee	writes errors to stderr (VMS/MS-DOS default)
Diagnostics_to=stdout	-Feo	writes errors to stdout (UNIX default)

3. The following statement should be added to the discussion of the branch tail merging optimization in the *Optimizations Chapter* of the *Reference Manual*:

   This optimization is tied to the **optimize=globalflow**(**-Og**) option.

4. (VMS only) The following step should be added to the section *Installation Procedure* of the *VMS Installation* chapter of the *Installation Guide:*

   11. The logical name **c68k$library:** must be predefined to locate libraries and standard include files. It is recommended that this be done system-wide in the **systartup.com** file by using the following command:

   ```
   $ assign/system sys$sysroot:[mcc68klib] c68k$library:
   ```

5. There are several errors in the *Embedded Environments* chapter of the *Reference Manual*. They are described as follows:

   Section *In-Line Assembly:*

   The statement in the subsection *Variable Names Inside **asm***

   > Variable names inside a string are not affected by compiler options like **prepend=dot** (VMS) or **-upd** (UNIX/DOS).

   conveys misleading information and should be changed to:

   > Variable names inside a string are not affected by compiler options like **prepend=dot** (VMS) or **-upd** (UNIX/DOS) unless they are enclosed in back quotes.

   There are several erroneous examples in the subsection *Other Examples*.

   Example:

   unsigned char save_D1=asm(unsigned char, " MOVE.L D1,D0");

   The compiler produces the following sequence:

   MOVE.L D1,D0

   MOVE.B D0,_save_D1

The second instruction is incorrect. D0 will actually be moved to the location of the local variable save_D1, say, a6-1, so the correct insturction should be:

```
MOVE.B D0,1(a6)
```

Example:

```
(void) asm (" CMP $80,_mask"," BEQ label", " TRAP #0",
"LABEL: ");
```

It should be changed to:

```
asm (void, " CMP $80,_mask", " BEQ label", " TRAP #0",
"LABEL: ");
```

Example:

```
asm(char*, " MOVEA.L #0,A0")[100]='*';
```

It should be changed to:

```
asm(char*, " MOVE.L #0,D0")[100]='*';
```

Example:

```
If (asm (struct {void *ret_addr; int parm1,parm2;}*;
        " MOVEA.L A6,A0 ") -> ret_addr<0x8FFFFFFF) . . .;
```

It should be changed to:

```
If (asm (struct {void *ret_addr; int parm1,parm2;}*;
        " MOVE.L A6,D0 ") -> ret_addr<0x8FFFFFFF) . . .;
```

Section *User-Modified Routines*:

If no INPORT command is issued, XRAY will read from the **standard I/O**. It should actually be **standard input.**

Section *Using the Linker*, subsection *INITDATA Command (IEEE Only)*:

The third paragraph should be replaced by the following:

The Microtec Research run-time libraries contain the **initcopy** function. If the **INITDATA** command is used, the initcopy function should be called from the **START** function in **CSYS.C**. The **CSYS.C** file on the distribution includes a call to **initcopy** which is commented out. If any modifications are made, **CSYS.C** must be recompiled with **-DINITCOPY** and the resulting object module must be linked with your program.

Installation

Flexible Licensing, Version 2.0

This product uses Flexible Licensing, Version 2.0, on the following hosts:

Apollo/HP DOMAIN/OS 10.3 or later
DECStation or DECSystem ULTRIX 4.2 or later
HP 9000/700 running HP-UX 8.0 or later
IBM RS/6000 AIX version 3.1 or later
Sun-3/SunOS 4.0 or later
Sun-4/SunOS 4.0 or later

For these hosts, refer to the *Flexible License Manager Documentation Set*, which contains an *Installation Guide* and *User's Guide*.

If you are using Microtec Research programs with Flexible Licensing Version 1, or programs from other vendors that use Highland Software's Flexible Licensing, you should read Appendix C, *Compatibility Issues*, in the *Flexible License Manager Installation Guide*.

Flexible Licensing Version 2.0 does not use **mricheckout** and **mricheckin**.

Name Expansion is Not Implemented

The name expansion capability of **mlicense**, described in the *Flexible License Manager User's Guide*, is not implemented in Version 2.0. If you want to check out licenses in advance, you must be careful to itemize all licenses you will need. Keep in mind that the MCC68K license requires the ASM68K license, since the compiler's output is usually run through the assembler, and you will need the linker (which requires ASM68K) as well. If you are debugging C code with XRAY, you need the XHS68K license. Here are examples of 24-hour advance checkouts:

```
mlicense -h 24 -g mcc68k asm68k
mlicense -h 24 -g mcc68k asm68k xhs68k
```

If you are compiling C++ code, you need CCC68K, MCC68K, and ASM68K. If you are debugging C++ code with XRAY, you need both XHS68K and XOS68K. Here are examples of 24-hour advance checkouts:

```
mlicense -h 24 -g ccc68k mcc68k asm68k
mlicense -h 24 -g ccc68k mcc68k asm68k xhs68k xos68k
```

You can also hold licenses for 24 hours (rather than 1 hour) after the last use by setting the following environment variable:

```
setenv MLICENSE_CHECKOUT_HOURS 24
```

Undocumented Features in Flexible Licensing

New Environment Variable: LICENSE_LOG_FILE

If you have floating licenses, the environment variable **LICENSE_LOG_FILE**, if set, overrides **/usr/tmp/license.log** (on each license server host) as the location of the log file.

Restart Script Can Send Electronic Mail

If you have floating licenses and the log file is not writable, the restarting script will open **/dev/null** as the log file and send you electronic mail to that effect.

Known Problems in Flexible Licensing

Temporary Loss of Licenses When Daemons Restart

If you have floating licenses and the daemons **lmgrd** or **MRI** go down, you may get the following message when requesting a new license (e.g., by running a different licensed program):

```
Giving up licenses and restarting mlicense.daemon.
```

Just run the programs you need, or type:

```
mlicense -g progname...
```

and you will get your licenses back.

Avoid Relative Paths

Automatic restarting of Highland daemons does not always work on automounting hosts if the executables are in a **PATH** element that is relative (e.g., **../mri/bin**). Use absolute directories (leading /) in your **PATH** or **path** environment variable definitions.

MS-DOS Host

DOS/16M Memory Management System

The DOS/16M memory management system is now used by MCC68K on IBM-MS-DOS host systems. It is transparent to the user except that on certain machines DOS/16M is unable to detect the type of machine that is running MCC68K. Therefore, the default settings set by DOS/16M do not work. In these special cases, you must set a DOS environment variable that will override the default settings. The DOS/16M environment variable must be set on the following machines as shown below:

NEC 98-series:

```
set DOS16M=1 @start_address-end_address
```

Fujitsu FMR-60, 70:

```
set DOS16M=5 @start_address-end_address
```

AT&T 6300+:

```
set DOS16M=6 @start_address-end_address
```

The *start_address* and *end_address* are numbers which can be expressed in decimal or hexadecimal format. Hexadecimal format requires a **0x** prefix. The number may end in a **K** or **k** to indicate an address in kilobytes, or an **M** or **m** to indicate megabytes. If no suffix is specified, the default is kilobytes. An example of setting the DOS16M environment variable on a NEC 9801 is shown below:

```
set DOS16M=1 @1m-2m
```

File Placement

UNIX

A complete list of files appears in the file *install_dir*/**mcc68k/mcc68k.fil** (*install_dir* is normally **/usr/mri**). The files are installed in the following directories:

Executables	*install_dir*/**bin**
Include files	*install_dir*/**include/mcc68k**
Library files	*install_dir*/**lib**
Support files	*install_dir*/**mcc68k**

MS-DOS

A complete list of files appears in the file *install_device*:\mcc68k\mcc68k.fil (*install_device* is normally the **C** drive). The files are installed in the following directories:

Executables	*install_device*:**mcc68k**
Include files	*install_device*:**mcc68k**
Library files	*install_device*:**mcc68k\68000**
	install_device:**mcc68k\68020**
Support files	*install_device*:**mcc68k**

VMS

The installation procedure automatically places distribution files in the following directories. For a complete list of distribution files, see the file **SYS$COMMON:[SYSHLP.EXAMPLES.MRI68K]mcc68k.fil.**

Executables	**SYS$COMMON:[SYSEXE]**
Include files	**SYS$COMMON:[MCC68KLIB]**
Library files	**SYS$COMMON:[SYSLIB]**
Support files	**SYS$COMMON:[SYSHLP.EXAMPLES.MRI68K]**

Search Paths

Please see the *User's Guide* chapter for UNIX/MS-DOS/VMS for information on environment variables that should be used if the distribution files are not installed in their default directories. See the section *Environment Variables*.

Also see the *User's Guide* for information on the search algorithm for include files (see options **ipath=*dir***(**-I**) and **spath=*dir***(**-J**)).

Release 4.2D

Enhancements in 4.2D

1. (SPR 13303) The compiler now generates more efficient code for bit-field operations. In some situations, bit-field instructions may not be generated even with the option **-p68020** specified because the improved alternate instruction sequence actually runs faster.
2. (SPRs 10100, 13765, 13789) More efficient code will now be generated for accessing volatile pointers.

Corrections in 4.2D

1. (SPR 13326)The compiler support file will no longer cause the link-time error message:

```
Fatal error 300 in module entry part: debug information
warning: BB10 block illegally nested in BB5.
```

2. The following code-generation problems have been fixed:
   - (SPRs 11491, 12641) If the address of a struct was added to an unsigned integer and the size of the struct was less than 32K bytes, then a 16-bit addition would be performed.
   - (SPR 11235) Incorrect code was generated to reference fp7 when there was no FPU.
   - (SPRs 10933, 11390, 11406, 12159, 12635) The compiler sometimes generated illegal instructions which resulted in assembly-time errors. These errors were due to the incorrect usage of **An** and **Dn** registers. For example, the following illegal instructions used to be emitted:

```
Instructions
lsl.l    #1,An
adda.b   Dn,An
jsr      (Dn)
move.b   An,Dn
bra.l    <label>  (for 68000 & 68010)
```

3. (MS-DOS only) (SPR 14157) Sometimes a General Protection Fault at 00D0:00E7 in CBE68KEM.EXE resulted when the options -B -Og -Fsm were used.

Release 4.2C

Corrections in 4.2C

1. The following code-generation problems have been fixed:
   - (SPR 13316) Incorrect code was generated for accessing bitfields in a structure: the higher byte was accessed instead of the lower byte.
   - (SPRs 13423, 13742, 13717, 13460, 14050, 14176) Compiler generated "`bcc.s L????`" instruction which failed to reach the label, causing an assembly-time error and the following error message to be generated:

     ```
     ERROR:(602) Calculated displacement does not fit in its
     field - truncated.
     ```
   - (SPRs 12509, 13461, 13700) Inline **memcpy** corrupted **Dn** register which was used for another program variable.
   - (SPR 13696) Incorrect code was generated for accessing bitfields in a structure. Bit-field was extracted into **D2**, but was referenced as though it was in **D1**.
   - (SPR 13714) Core dump problem when passing struct by value.
   - (SPR 13764) With inline **asm**, the option **reserve=An**(-**Mr**<*n*>) to reserve a register did not work.
   - (SPR 13814) An EXTERN struct passed by value was not aligned on the stack properly; as a result, the called function got incorrect values.

Release 4.2B

Corrections in 4.2B

1. The following code-generation problems have been fixed:
   - (SPRs 10794, 11823) Incorrect code was generated when indexing into an array < 32K with expression involving constant subtraction.
   - (SPR 12088) If a struct was an argument and was passed by value to another function, it was not aligned on the stack properly; as a result, the called function got incorrect values.
   - (SPRs 12607, 12941) In 68040 mode the compiler generated the "**fintrz**" instruction. This instruction is not available on the 68040. This would cause the program to trap at run-time.

Release 4.2A

Operational Changes in 4.2A

1. (UNIX/MS-DOS only) The **debug=restricted(-Gr)** and **debug(-g)** options are now mutually exclusive.
2. Stack and heap sections are now created in the supplied start-up code. The length of these sections may be set at link time. **???STACKTOP** is no longer used in the supplied start-up code.
3. Any source file that was compiled with a previous version of the compiler and includes **stdio.h** must be recompiled with the 4.2A MCC68K compiler.
4. The file list is now available online.
5. The MMER (merge utility program) and XRFC (cross reference utility program) are no longer distributed.
6. The **pointer_init(-Kp)** option has been removed.
7. The **rename=(data=*name*)(-ND*name*)** option has been changed to **rename=(zerovars=*name*)(-NZ*name*)**. If **rename=(data=*name*)(-ND*name*)** is used, a warning message will be issued.

```
(UNIX)      (W) option -ND is being phased out; use -NZ
```

8. The **debug=multi_stmt(-Gm)** option is now the default.
9. The branch tail merging optimization is now tied to **optimize=globalflow(-Og)**.
10. The **-p68332** option has been changed to **-pcpu32**. The option **-pcpu32** only works with Version 2.2 (or later) of XRAY68K; earlier versions would only recognize **-p68332**.
11. The following 4.1-style options are no longer accepted in Release 4.2A. A warning message will be issued if they are used. Instead, the user should use the new 4.2A options shown below.

4.1 Options	4.1B/4.2A Options	
-Fle	-Fli	Show contents of #include files in listing
-Fls	-nFli	Don't show contents of #include files in listing

12. A different set of RTL support files is shipped with the 4.2A MCC68K compiler. Please refer to the **mcc68k.fil** file for the file list. If more information is desired, please read the source files.
13. **puts**() used to write "\r\n" at the end of a string. It now simply writes "\n".
14. **fopen**() now calls **open**() with UNIX-like flags.
15. Additional static data used by libraries has been moved to **csys.c**.
16. Math functions will now set **errno** when a denormalized number is generated.

17. **_fp_control** and **_fp_status** are no longer used by math library.
18. **csys68k.c** is now **csys.c**. System level routines which were located in **csys68k.c** are now located in individual C source files. Modifications were made to **csys.c** improve ANSI conformance.
19. **csys.c** now also opens **stdaux** and **stdprn** in addition to **stdin**, **stdout**, and **stderr**.
20. Parameter names have been removed from all function prototypes in the Microtec Research standard include files.
21. On UNIX hosts, errors used to be written to stderr, they are now written to stdout. (See also: **-diagnostics_to=stderr(-Fee)/diagnostics_to=stdout(-Feo)** option)

Enhancements in 4.2A

1. Bitfield instructions are now used in 68020 mode.
2. The option **-pcpu32** has been added to support the CPU32 family of processors.
3. Support for the MC68030 and MC68040 has been added. Instruction scheduling optimization has been added for the 68040 processor.
4. Instruction scheduling optimization has been added for the 68881 and 68882 processors.
5. Better code is now generated for 1-bit bitfields. The BCLR/BSET instructions are now used instead of AND and OR.
6. Debug information is now generated for inlined functions.
7. Occasionally, the compiler was generating an unnecessary save/restore of FP2.
8. Several compiler limits have been expanded or removed:
   - There is no longer any limit on initializer lists.
   - There is no longer any limit on the number of tokens per function.
   - The limit on the size of strings has been raised from 1024 characters to 2048.
   - The number of case labels allowed in one switch statement has been raised from 512 to 8192.
9. Structures can now be initialized on their declaration.
10. (SPRs 8076, 8676, 8973, 9208) You may now redirect errors by using the **diagnostics_to=stderr(-Fee)/diagnostics_to=stdout(-Feo)** option.
11. The **nomri_extensions(-nx)** option is now doing a better job of diagnosing non-ANSI Microtec extensions.

12. The compiler will now generate additional debug information to allow the debugging of multiple-statement lines. This feature will only work with Version 2.2 (or later) of XRAY68K. If you wish to disable this feature, you must use the **debug=nomulti_stmt(-nGm)** compilation option.

13. (SPR 6146)The **asm** feature has been improved. You can now use variable names within **asm** strings. Please see the *MCC68K Reference Manual* for more information.

14. New predefined symbols:

```
_PIC, _PID (-Mcp, -Mdp, -Md<n>)
_CHAR_[UN]SIGNED (-Ku, -nKu)
_MRI_EXTENSIONS (-x, -nx)
```

15. New Options:

UNIX/MS-DOS Options	VMS Options
-p68030	/cpu=68030
-p68040	/cpu=68040
-Gl**	/debug=lines**
-Oi	/optimize=inline
-Ol	/optimize=local
-Or	/optimize=reorder
-Oe	/optimize=singlereturn
-ui<*char*>	/insert_char=<*char*>
-ut<*number*>	/truncate_identifiers=<*number*>
-O	not available on VMS
-Fee	/diagnostics_to=stderr
-Feo	/diagnostics_to=stdout

(**The new option addresses SPR 6895, SPR 9485, SPR 9972, SPR 10696, and SPR 10714)

16. The runtime libraries have been made more re-entrant. Also, all non-re-entrant library routines are now documented. Please refer to the *MCC68K Reference Manual* for detailed information.

17. **qsort**() is now reentrant.

18. The file **entry.s** has been renamed to **entry.c**. It has been modified to improve readability and ANSI conformance.

19. Improved overflow handling in **strtol**().

20. **putw**() and **putl**() now write in 68000 memory order. In the past they wrote in reverse order. **getw**() and **getl**() now read in 68000 memory order. In the past they read in reverse order.

21. (SPRs 9670, 9761, 9763, 9670, 9876, 10657, 10668, 10709) New RTL Functions:

fsetpos	memmove	difftime	rewind
asctime	setlocale	unlink	clock
strerror	vprintf	atexit	remove
zalloc**	bsearch	time	localtime
strtoul	strstr	vsprintf	gmtime
vfprintf	ctime	div	strcoll
ldiv	ftell	fgetpos	perror

**zalloc was mistakenly omitted in Release 4.1e

Corrections in 4.2A

1. In some instances, **optimize=globalflow(-Og)** was producing less optimized code than **optimize=noglobalflow(-nOg)**. This has been corrected.
2. A diagnostic message is now generated for references to undefined enums.
3. Formerly, **data_addressses=anrelative(-Mdn)**, **data_addresses=pcrelative(-Mdp)** and **reserve=a*n*(-Mr)** were diagnosed as being conflicting options. In release 4.2A, only **data_addressses=anrelative(-Mdn)** and **reserve=a*n*(-Mr)** are diagnosed as conflicting options.
4. Floating-point overflow and underflow are now diagnosed for floating-point constants. If a float data item is less than MIN_FLOAT, or greater than MAX_FLOAT, it will be set to 0 or MAX_FLOAT respectively. If a double data item is less than MIN_DOUBLE, or greater than MAX_DOUBLE, it will be set to 0 or MAX_DOUBLE respectively. An appropriate diagnostic message will be generated.
5. The following code-generation problems have been fixed in this release:
   - Incorrect code was being generated for "`expr ? p=<e> : *p = <e>`"
   - Passing a structure greater than 64k in size as a function parameter was resulting in an assembly-time error.
   - Passing the address of a local array greater than 32k in size as a function parameter was resulting in an assembly-time error.
   - The code generated for accessing bitfields was incorrect if the bitfields were declared as volatile.
   - The code generated for a call through a pointer to a function which was a candidate for being in-lined was resulting in an assembly-time error.
   - The **optimize=globalflow(-Og)** and **optimizer=stablemem(-Ob)** options, when used together, sometimes resulted in a temporary variable being used incorrectly.
   - When **optimize=globalflow(-Og)** and **reserve=a2(-Mr2)** were used together, occasionally the wrong register would be saved/restored.
   - Incorrect code was being generated for "`f(strcpy(a,b))`".

- The optimizer was occasionally removing necessary loads of a volatile data item.
- (SPRs 9713, 9731) Code Generation performed longword read from an odd address causing 68000 trap.
- (SPR 10322) Incorrect code was being generated for assignment to parameters in a loop statement.
- (SPRs 10479, 11663, 11824) A 16-bit value was being stored into a 32-bit register without clearing the upper 16 bits.
- (SPRs 10560, 10643) Incorrect code was sometimes generated when 64K byte array was indexed using unsigned short.
- (SPR 10802) When a structure was greater than 32K and the **optimize=space**(**-Os**) option was used, the array indices were forced to be computed in short rather than in long.
- (SPR 11510) Generated incorrect code

  ```
  move.b (a1,d0.w),(a0,d0.l)
  ```

  instead of

  ```
  move.b (a1,d0.w),(a0,d0.w)
  ```
- (SPR 11711) The compiler was incorrectly using 16-bit indexing for referencing a 40,000-byte array.
- (SPR 11768) Incorrect code which destroyed the register contents was generated with **optimize=register**(**-OR**) (default).
- (UNIX/MS-DOS only) (SPRs 11815, 12145) A General Protection Fault at 0128:104F in CBE68KEM.EXE on the MS-DOS host or a core dump on UNIX resulted when the options -g -B -Og were used.
- (SPR 11979) The error "`(print_oe_secondary_state) Error in pname op 8`" was issued when running CBE68K.
- (SPR 12002) Incorrect count was generated in a for loop with **optimize=globalflow**(**-Og**).
- (MS-DOS/VMS/HP Only) (SPRs 10563, 8064, 10606) Improper debug info led to displaying wrong values for variables when running XRAY.
- (MS-DOS Only) (SPRs 8600, 8703) Getting error: DOS/16M ERROR: Program must be built with -auto for DPMI.
- (MS-DOS Only) (SPRs 8382, 10279) When using **-g**, option, got message: compilation terminated, not enough memory proc/dcl too large.
- (SPR 10919) Incorrect code was sometimes generated with the **-g** option.
- (SPR 12015) Incorrect code was generated which resulted in an assembler error of an illegal operand in a move instruction.

6. On some occasions, incorrect debug information was being generated for register variables and frameless functions. This has been corrected.
7. The library routine **fopen()** was starting file allocation with fileid=5 rather than fileid=0. This has been corrected.
8. The **quit=number(-Q<*n*>)** option was printing <*n*+2> error messages, rather than <*n*>.
9. The fields of a struct declared as *volatile* were not being treated as *volatile*. This has been corrected.
10. (SPRs 8076, 8676, 8973, 9208) Unable to redirect error output.
11. (SPR 9947) Formerly, when using **-Mdp** and **-Mr5** together, the driver would issue a warning message.
12. (SPR 10840) Incorrect code was sometimes generated when **optimize=globalflow**(**-Og**) was used.
13. (SPR 10869) size_t is now defined in both **stddef.h** and **stdlib.h**.

Using MCC68K 4.2X with XRAY prior to 2.2

1. The 4.2X compiler now generates additional debug information to allow the debugging of multiple-statement lines. This new feature is, however, not supported in XRAY68K prior to Version 2.2. You will have to use the **debug=nomulti_stmt(-n-Gm)** compilation option when using 4.2X MCC68K with an earlier version of XRAY68K.
2. The **-p68332** option has been changed to **-pcpu32** in the 4.2X compiler. This chip type will not work with versions of XRAY68K prior to 2.2. In order to make it feasible to use 4.2X MCC68K with earlier versions of XRAY68K, the **patch68k** utility has been provided to patch the chip type. For a more detailed description on how to utilize the **patch68k** utility, please read the Section *Using 68K Tools with XRAY* in the *Release Notes* for Microtec Research ASM68K Version 6.8D.

Known Problems

1. The library functions **scanf()**, **fscanf()**, and **sscanf()** may return an incorrect value for the number of input items assigned if a matching or input error occurs.
2. The **%i** format of the **scanf()** library routine is not implemented.
3. The **trace(-Kt)** option will cause link-time duplicate publics if two modules use the same name for static functions.
4. If a runtime library function which could be in-lined (like **memcpy**, **strcpy**, etc.) is invoked with the wrong number of arguments, the compiler may abort.
5. (UNIX/MS-DOS only) If there is a space between the **-e** option and its parameter, the compiler will abort.
6. No warning is issued if the user specifies both **optimize=nolocal(-nOl)** and **optimize=combine_pops(-Oc)**.
7. If **optimize=(globalflow,reorder,inline)(-O)** and **show=source(-Fsm)** have been specified, the intermixed C and assembler code in the output listing for large switch statements may not be correct.
8. (UNIX/MS-DOS only) The **-O** option will automatically invoke the function-inlining optimization. If you are going to debug the resulting absolute file using the XRAY68K debugger, you should specify **-O -nOi** to specifically disable the function-inlining optimization.
9. The **literals_addr_as=code** (**-alc**) and **literals_addr_as=data** (**-ald**) options should control how items in the literals section are referenced: according to the **code=** (**-Mc**) or **data=** (**-Md**) options. Instead, items in the literals sections are affected by the **string_addr_as=code** (**-acs**) and **string_addr_as=data** (**-asd**) options. The **literals_addr_as=code** (**-alc**) and **literals_addr_as=data** (**-ald**) options are ignored.
10. (MS-DOS only) The compiler does not generate an error message if the disk is full. Instead, it creates an empty .src file.
11. (MS-DOS only) Due to a problem in the Rational memory management system, if ^C is entered while the driver is invoking one of the passes, the system may hang. The user must re-boot the system.
12. (MS-DOS only) Due to a *<ctrl-c>* handling problem in the **MCC68K** driver, occasionally, one of the following run-time errors may be generated before exiting to **DOS**.

    ```
    run-time error R6000
    -stack overflow
    ```

    or

    ```
    VM error[307]: segment fault recursion
    ```

    Note that when this happens, temporary files will be left in the current directory or in the **TMP** directory. These files should be deleted.

13. Data structures larger than 64K bytes are rounded up to the next 256 byte boundary. For example, an array declared as 100000 bytes will actually be allocated 100096 bytes. Correct code is generated for accessing the array, but some data space is wasted.

14. (VMS only) The **-D** and **-U** options do not work with **#pragma options**. To work-around this problem, use the command line options **define** and **undefine** to define or undefine preprocessor symbols.

15. (68000, 68010, CPU32 only) In some cases the compiler will generate illegal code for accessing word- and longword-sized data allocated at odd addresses. The illegal code will cause the user program to trap at run-time. This problem is most often triggered by references to fields at odd offsets within packed structures, but can also be triggered by certain structure and string copy operations. In most cases, the problem can be worked around by disabling global optimization for the effected modules.

16. (Fixed in Release 4.2D)The code generated for accessing volatile pointers is not as efficient as it could be. When a volatile pointer is dereferenced, the compiler will generate code to move the value of the volatile pointer to a scratch stack location. This is slightly wasteful of code space (unnecessary move instruction) and of stack space.

17. Assignment to bitfield sets the location to zero first, then sets the value. For example, "`x.a=2`" gets "`andi.b #63,x.a, ori.b #128,x.a`". This causes an i/o output of zero before setting the bit.

18. The compiler will generate the error message "`symbol table too large for intermediate file`" for symbol tables greater than 64K. A workaround is to break down the module into smaller pieces.

19. (Fixed in Release 4.2B) In some cases the compiler will generate incorrect code for indexing into arrays of less than 32K bytes. This may occur if the index expression involves subtraction by a constant, and the index expression is a common subexpression.

20. (Fixed in Release 4.2D)The compiler will sometimes generate illegal instructions, resulting in assembly-time errors. The following shows the illegal instructions and the corresponding error messages generated by ASM68K:

Instructions	**Error Messages**
`lsl #1,An`	`This combination of operands is not valid for this instruction.`
`adda.b Dn,An`	`Invalid opcode/qualifier combination.`
`jsr (Dn)`	`This combination of operands is not valid for this instruction.`

21. (SPRs 14101,14521)The compiler sometimes generates illegal instructions, which incorrectly use **An** register, if the left side of += or -= is a pointer and the right side is an expression composed of a simple addition of 2 integers allocated in registers. For example,

```
ptr += int1 + int2;
```

will emit the illegal instruction "sub.l An,<ea>", which results in the following assembly-time error:

```
This combination of operands is not valid for this instruc-
                tion.
```

To work around this problem, use:

```
ptr = ptr + (int1 + int2);
```

22. When a loop counter is not initialized in a **for** statement, the compiler would core dump.

23. (SPR 14001) The compiler sometimes generates code to do 16-bit multiplication for integers if the left side of a += or -= is a pointer. For example,

```
char *z;
int x,y;
z += x * y;
```

To work around this problem, use:

```
z = z + x * y;
```

24. In 68881 mode, if a 'double' which is allocated in a register is assigned to a 'float' which is also allocated in a register, the value of the 'double' will not be truncated before it is assigned to the 'float'.

25. In some cases, the address of a static volatile variable cannot be used as a constant address. For example:

```
static volatile int a[2];
static int  *p = &a[1];/* (E)invalid initialization for "p" */
```

To work around this problem, use:

```
static in *p = (a +1);
```

26. When consecutive lines of declarations contain syntax errors, the compiler will generate incorrect error messages. For example,

```
int i 15;
int j 1;
int k j+2;
```

instead of complaining about '15', '1', and 'j', the following messages will be issued:

```
<file>, <line>; (E) syntax error; unexpected symbol: "15"
<file>, <line>; (E) syntax error; unexpected symbol: "k"
<file>, <line>; (E) syntax error; unexpected symbol: "j"
```

27. (SPR 14508) The library function **ftoa** sometimes overwrites the buffer passed to it.

28. If an expression that will cause an overflow condition is compared as being "less than" zero, the MC68000 condition codes may not get set correctly.

29. Debugging of **for** and **while** loops may not behave as expected when the **debug=nomulti_stmt** (**-nGm**) and **opt=global** (**-Og**) options are used. The **debug=multi_stmt**(**-Gm**) option is the default.

    If the loop is known to execute at least once, a breakpoint set on the line containing the **for** or **while** statement will not be hit prior to the first invocation of the loop. The breakpoint is actually placed on the code which performs the compare operation. Since the loop is known to execute at least once, the compare operation is not performed prior to the first execution of the loop. Therefore, when the breakpoint is first hit, the loop has already been executed once. The breakpoint will be hit on all subsequent loop executions.

    Example 1:

```
for (i = 0; i < 5; i++)/* line 10 */
{ ... body of loop ...
}
```

    This **for** loop is known to execute at least once (since the upper bound is a constant (5), and it is greater than the initialization value (0)). If a breakpoint is set on line 10, the first time the breakpoint is hit, the loop will have already executed once and the value of *i* will be 1. The code to perform the compare is about to execute.

    Example 2:

```
i = 5;
while (i)          /* line 15 */
{ ... body of loop ...
}
```

    This **while** loop is known to execute at least once (since *i* is known to be non-zero). If a breakpoint is set on line 15, the first time the breakpoint is hit, the loop will have already executed once, and the code to perform the compare is about to execute.

    To work-around this problem, try one of the following:

    1. Set the breakpoint on the first line of the body of the loop.

       Note that if the loop executes zero times (the condition evaluates to false before the first iteration), the breakpoint on the first line of the body will not be hit.

    2. Use the **opt=noglobal** (**-nOg**) option to disable this (and other global) optimizations. The **opt=noglobal** option is the default.

3. For a **for** loop, move the compare and increment statements to the line following the initialization statement:

```
for (i = 0;             /* line 10 */
   i < 5; i++)/* line 11 */
```

Microtec Research Inc.

2350 Mission College Blvd.

Santa Clara, CA 95054

Tel. 408.980.1300

Toll Free 800.950.5554

FAX 408.982.8266